Listing of Claims

1-66. (Cancelled)

67. (New) A computer implemented method for adding tamper resistance to a software program, the method comprising:

adding a silent guard variable to the software program;

selecting a computation in the software program;

determining an expected value of the silent guard variable at the execution point of the selected computation; and

revising the selected computation to be dependent on the runtime value of the silent guard variable, such that the selected computation will evaluate improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable.

68. (New) The method of claim 67, further comprising:

selecting a program variable in the software program;

determining an expected value of the selected program variable at a dependency point in the software program; and

making the runtime value of the silent guard variable dependent on the runtime value of the selected program variable at the dependency point.

69. (New) The method of claim 68, wherein the step of making the value of the silent guard variable dependent on the runtime value of the selected program variable comprises:

computing the runtime value of the silent guard variable using a mathematical expression including the runtime value of the selected program variable and the expected value of the selected program variable at the dependency point.

70. (New) The method of claim 67, wherein the step of revising the selected computation comprises:

selecting a constant value used in the selected computation; and

replacing the constant value with a mathematical expression that is dependent on the runtime value of the silent guard variable, such that the mathematical expression evaluates to the constant value if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable.

71. (New) The method of claim 67, wherein the step of revising the selected computation comprises:

selecting a computation variable used in the selected computation;

determining an expected value of the computation variable at the execution point of the selected computation; and

replacing the computation variable with a mathematical expression that is dependent on the runtime value of the silent guard variable, such that the mathematical expression evaluates to the expected value of the computation variable if the runtime value of the silent guard variable is equal to the expected value of the silent guard variable.

72. (New) The method of claim 67, wherein the step of revising the selected computation comprises:

inserting a mathematical expression including the runtime value of the silent guard variable and the expected value of the silent guard variable into the selected computation.

73. (New) A computer implemented method for adding tamper resistance to a software program, the method comprising:

selecting a program variable in the software program;

determining an expected value of the program variable at a first dependency point in the software program; and

adding a silent guard variable to the software program;

determining an expected value of the silent guard variable at the first dependency point; and

making the runtime value of the program variable dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will equal the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable equals the expected value of the silent guard variable at the first dependency point.

74. (New) The method of claim 73, wherein the step of making the runtime value of the program variable dependent on the runtime value of the silent guard variable comprises:

initializing the program variable to equal the runtime value of the silent guard variable at the first dependency point.

75. (New) The method of claim 73, wherein the step of making the runtime value of the program variable dependent on the runtime value of the silent guard variable comprises:

installing a mathematical computation that includes the runtime value of the silent guard variable, such that the result of the mathematical computation is corrupted if the runtime value of the silent guard variable is not equal to the expected value of the silent

guard variable at the first dependency point; and

initializing the program variable to equal the result of the mathematical computation.

76. (New) The method of claim 73, wherein the runtime value of the program variable changes during execution of the software program, comprising:

determining an expected value of the program variable at a second dependency point in the software program; and

adding a supplementary silent guard variable to the software program;

determining an expected value of the supplementary silent guard variable at the second dependency point; and

making the runtime value of the program variable dependent on the runtime value of the supplementary silent guard variable, such that the runtime value of the program variable will equal the expected value of the program variable at the second dependency point if the runtime value of the supplementary silent guard variable equals the expected value of the supplementary silent guard variable at the second dependency point.

77. (New) The method of claim 73, wherein the runtime value of the program variable changes during execution of the software program, comprising:

determining an expected value of the program variable at a second dependency point in the software program; and

determining an expected value of the silent guard variable at the second dependency point; and

making the runtime value of the program variable dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will equal

the expected value of the program variable at the second dependency point if the runtime value of the silent guard variable equals the expected value of the silent guard variable at the second dependency point.

78. (New) A computer implemented method for adding tamper resistance to a software program, the method comprising:

selecting a program variable in the software program;

selecting a computation in the software program;

determining an expected value of the program variable at the point of execution of the selected computation; and

revising the selected computation to be dependent on the runtime value of the program variable, such that the selected computation will evaluate incorrectly if the runtime value of the program variable is not equal to the expected value of the program variable.

79. (New) The method of claim 78, wherein the step of revising the selected computation comprises:

selecting a constant value used in the selected computation; and

replacing the constant value with a mathematical expression that is dependent on the runtime value of the program variable, such that the mathematical expression evaluates to the constant value if the runtime value of the program variable is equal to the expected value of the program variable.

80. (New) The method of claim 78, wherein the step of revising the selected computation comprises:

selecting a computation variable used in the selected computation;

determining an expected value of the computation variable at the point of execution of the selected computation; and

replacing the computation variable with a mathematical expression that is dependent on the runtime value of the program variable, such that the mathematical expression evaluates to the expected value of the computation variable if the runtime value of the program variable is equal to the expected value of the program variable.

81. (New) The method of claim 78, wherein the step of revising the selected computation comprises:

inserting a mathematical expression including the runtime value of the program variable and the expected value of the program variable into the selected computation.

82. (New) A computer implemented method for adding tamper resistance to a software program, the method comprising:

selecting a program block containing a step necessary for proper execution of the software program, the program block comprising at least one program instruction;

selecting a silent guard for the program block;

determining the expected value of the silent guard at the start of execution of the program block; and

installing a branch instruction dependent on the silent guard in the software program, such that if the runtime value of the silent guard is not equal to the expected value of the silent guard then the branch instruction will cause an incorrect branch to be taken.

83. (New) The method of claim 82, wherein the step of selecting a silent guard comprises:

adding a silent guard variable to the software program;

using the silent guard variable as the silent guard; and

installing an initialization instruction for the silent guard variable that executes prior to the branch instruction in the software program, the initialization instruction setting the silent guard variable equal to the expected value of the silent guard.

84. (New) The method of claim 82, wherein the step of selecting a silent guard comprises:

selecting a program variable in the software program; and

using the program value as the silent guard.

85. (New) The method of claim 82, wherein the step of selecting a silent guard comprises:

selecting an insertion point in the software program;

selecting a program variable in the software program;

determining the expected value of the program variable at the insertion point;

making the runtime value of the silent guard dependent on the runtime value of the program variable, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point.

86. (New) The method of claim 85, wherein the step of making the runtime value of the silent guard dependent on the runtime value of the program variable, comprises:

installing a mathematical computation that includes the runtime value of the program variable, such that the result of the mathematical computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point; and

setting the silent guard equal to the result of the mathematical computation.

87. (New) The method of claim 86, wherein the silent guard computation uses both the expected value of the program variable and the runtime value of the program variable.

88. (New) A recordable computer media having a tamper resistant software program recorded thereon, comprising:

a software program comprising one or more program instructions;

a program computation in the software program at an execution point;

a silent guard variable in the software program having an expected value at the execution point;

wherein the program computation is dependent on the runtime value of the silent guard variable, such that the program computation will evaluate improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable.

89. (New) The recordable computer media of claim 88, further comprising a program variable in the software program having an expected value at a dependency point in the software program; wherein the runtime value of the silent guard variable is dependent on the runtime value of the program variable at the dependency point.

90. (New) The recordable computer media of claim 89, wherein the runtime value of the silent guard variable is also dependent on the expected value of the program variable at the dependency point.

91. (New) A recordable computer media having a tamper resistant software program recorded thereon, comprising:

a software program comprising one or more program instructions;

a program variable having an expected value at a first dependency point in the software program; and

a silent guard variable having an expected value at the first dependency point;

wherein the runtime value of the program variable is dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable equals the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable equals the expected value of the silent guard variable at the first dependency point.

92. (New) The recordable computer media of claim 91, further comprising an initialization instruction initializing the program variable to equal the runtime value of the silent guard variable at the first dependency point.

93. (New) The recordable computer media of claim 91, further comprising a mathematical computation that includes the runtime value of the silent guard variable, the program variable being dependent on the result of the mathematical expression; wherein the result of the mathematical computation is corrupted if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable at the first dependency

point.

94. (New) The recordable computer media of claim 91, further comprising a supplementary silent guard variable having an expected value at a second dependency point in the software program;

wherein the expected value of the program variable at the second dependency point is not equal to the expected value of the program variable at the first dependency point; and the runtime value of the program variable at the second dependency point is dependent on the runtime value of the supplementary silent guard variable, such that the runtime value of the program variable equals the expected value of the program variable at the second dependency point if the runtime value of the supplementary silent guard variable equals the expected value of the supplementary silent guard variable at the second dependency point.

95. (New) The recordable computer media of claim 91, wherein the expected value of the program variable at the second dependency point is not equal to the expected value of the program variable at the first dependency point; and the silent guard variable has a second expected value at the second dependency point; and the runtime value of the program variable at the second dependency point is dependent on the runtime value of the silent guard variable at the second dependency point, such that the runtime value of the program variable equals the expected value of the program variable at the second dependency point if the runtime value of the silent guard variable equals the expected value of the silent guard variable at the second dependency point.

96. (New) A recordable computer media having a tamper resistant software program recorded thereon, comprising:

a software program comprising one or more program instructions;

a program block containing a step necessary for proper execution of the software program, the program block comprising at least one program instruction;

a silent guard in the software program having an expected value at the start of execution of the program block; and

a branch instruction in the software program dependent on the silent guard, wherein the branch instruction will cause an incorrect branch to be taken if the runtime value of the silent guard is not equal to the expected value of the silent guard.

97. (New) The recordable computer media of claim 96, wherein the silent guard is a silent guard variable, the silent guard variable being set equal to the expected value of the silent guard prior to the branch instruction in the software program.

98. (New) The recordable computer media of claim 96, further comprising a program variable having an expected value at an insertion point; wherein the runtime value of the silent guard is dependent on the runtime value of the program variable, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point.

99. (New) The recordable computer media of claim 98, wherein the runtime value of the silent guard is made dependent on the runtime value of the program variable using a mathematical computation that includes the runtime value of the program variable, wherein the result of the mathematical computation is corrupted if the runtime value of the program variable

is not equal to the expected value of the program variable at the insertion point.

100. (New) The recordable computer media of claim 98, wherein the runtime value of the silent guard is also dependent on the expected value of the program variable at the insertion point.